

**Amendments to the drawings,**

*There are no amendments to the Drawings.*

## Remarks

### Status of application

Claims 1-49 were examined and stand rejected in view of prior art. The claims have been amended to further clarify Applicant's invention; however, these amendments to the claims are not intended to narrow the scope of Applicant's invention. Applicant has also amended claims 22, 42, and 43 for improving readability of these claims. Reexamination and reconsideration are respectfully requested.

### The invention

Applicant's invention comprises a system and methodology for improved optimization of a database queries including join conditions between two or more database tables. Applicant's invention provides for improved optimization of a database query by creating "subplans" within each query block (i.e., each atomic portion) of the query based on grouping together portions of each query block and then optimizing each of these subplans independently. Significantly, Applicant's invention provides for creation of "subplans" (subplan objects) which are utilized to group together quantifiers (e.g., quantifier objects defined for each derived table, base table, view, or subquery) used in outer joins of each query block. This technique of grouping together quantifiers of the outer joins enables improved optimization of the database query while requiring minimal memory.

After subplan objects have been created inside each query block, Applicant's invention independently optimizes each such subplan to determine a favorable access plan for each subplan (e.g., based on estimated execution costs and other properties). Applicant's optimization methodology includes examination of alternative join methods (i.e., physical join operators) and access methods (e.g., indexes) in determining a favorable access plan for each such subplan and not merely the join strategy or join order for joining relations.

The favorable access plan determined for each subplan of a query block is subsequently used in generating an optimal access plan for the query block. The resulting access plans which are obtained are naturally bushy in the presence of outer joins. A plan

for execution of the database query is then constructed based upon the optimal access plan determined for each query block.

#### General

##### A. Double patenting rejection

Claims 1-7, 10, 16-17, 22-25, 36-37, 40-43, 45, and 49 stand provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1-22 of Applicant's later-filed co-pending Application No. 10/835,230 (SYB/0107.00). Applicant does not agree with the Examiner's position, as it is believed that the two claimed inventions are patentably distinct. As the rejection is provisional, however, Applicant will defer discussion/argument concerning the issue until such time, if any, that the issue becomes ripe (i.e., allowed claims are obtained in at least one of the two applications).

##### B. Non-statutory subject matter rejection

Claim 25 stands rejected under 35 U.S.C. 101 on the basis that the claimed invention is directed to non-statutory subject matter. Here, the Examiner states that "claim 25 is directed towards a downloadable set of computer-executable instructions, which is merely an arrangement of data and therefore considered non-statutory." The claim has been amended to restate the claimed subject matter in terms of a process or method of operation. Accordingly, the rejection on the basis of non-statutory subject matter is overcome.

##### C. 35 U.S.C. 112, second paragraph rejection

Claim 9 stands rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Claim 9 states that all sets of "outer references" must have an access plan defined in the optimization process. The Examiner asserts that, although the term "outer references" appears in the specification, it appears only within algorithms or in reference to its use in those algorithms and is never defined. For these

reasons the term "outer references" is considered to be indefinite. The Examiner states that for the purpose of further examination the Examiner takes the term "outer references" to mean tables which are involved in an outer join.

The term "outer reference" is one that is well understood in the database arts. For example, the term is defined in Section 6.4 of the SQL ANSI standard ANSI/ISO/IEC 9075: 1992, which is incorporated by reference into Applicant's Specification (Applicant's Specification, page 4, lines 12-15). This SQL ANSI standard provides that an "outer reference" is a column reference used in a table expression which does not contain the column's table. In Applicant's Claim 9, outer references refer to column references used in a subplan whose tables are not tables in the subplan. For example, in Example 7 at page 23, line 24 to page 24, line 15 of Applicant's Specification, the subplan s1 refers to the outer reference "C\_Custkey" which is a column of the table "Customer" which is not in the subplan s1. Although Applicant believes that the term "outer reference" is well known by those skilled in the database arts, Applicant has amended claim 9 to replace the term "outer references" with "column references used in a subplan whose tables are not tables in the subplan", thereby overcoming the Examiner's objection.

Claim 19 stands rejected under 35 U.S.C. 112, second paragraph, for indefiniteness. The Examiner contends that "pipeline characteristics" is insufficiently characterized in the specification to permit its use in the claims. The Examiner states that for the purpose of further examination the Examiner takes the term "pipeline characteristics" to mean the order in which the operations of the execution strategy will be executed, e.g., the order in which the operations will enter the pipeline.

As described below in detail, Applicant's optimization methodology includes selecting particular "join methods", which are physical operators implementing logical joins. In generating access plans Applicant's invention selects particular physical operators to implement particular logical joins based on estimated execution costs and other properties. "Pipeline characteristics" is one of these properties and refers to the way the result set of a physical operator is computed (rather than the order of the joins: very different join orders may actually have the same pipeline characteristics). For example, a join physical operator is said to be "pipelined" if it is not input independent as described

in Definition 8 at page 29, line 20 to page 30, line 17 of Applicant's Specification. A join can, for instance, be implemented by the physical operator "Join Nested Loops" (JNL) which is a pipelined operator. A non-pipelined operator, in contrast, is a blocking operator which does not return any results until it has consumed at least one of its inputs. The join physical operators which are input independent are non-pipelined. For example, the physical operator Join Hash (or Hash Join) must see all the rows from one of the inputs to create a hash table before a row in the output is returned. Thus, the "pipeline characteristics" of an access plan is a physical property of the access plan describing if the rows of the result are computed in a pipeline manner by the access plan. Applicant has amended claim 19 to indicate that the pipeline characteristics of the access plan describe how results are computed by the access plan, thereby overcoming the objection.

Prior art rejections

A. Section 102(b) rejection: Bowman et al.

Claims 1-25, 36-40, and 42-49 stand rejected under 35 U.S.C. 102(b) as being anticipated by Bowman et al. ("Join Enumeration in a Memory-Constrained Environment", Proceedings, 16th IEEE Data Engineering Conference, San Diego, California; March 2000, referred to hereinafter as "Bowman"). The Examiner rejection of Applicant's Claims 1, 24, and 25 as follows is representative of the rejection of Applicant's claims as anticipated by Bowman:

As per Claims 1, 24, and 25, Bowman discloses in a database system, a method for constructing an optimal query execution plan for executing a query (i.e. "In today's computing environment, database technology can be found on virtually any device... The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space." The preceding text excerpt clearly indicates that a method for optimizing a query in a database system is presented.) (Abstract), the method comprising: receiving a query specifying at least one join condition between two or more database tables (i.e. "The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space." The preceding text excerpt clearly indicates that the query has a high join degree (e.g. specifies at least one join between two or more database tables).) (Abstract); identifying each query block within said query, each

query block comprising an atomic portion of said query (i.e. "The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1). A QOG is the internal representation of a complete SQL statement<sup>2</sup>, possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG." The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query, which each subquery/query block representing an atomic portion of the query (e.g. tables, predicates, etc.) (Page 2, Column 2, Paragraph 4); creating subplans for each query block based on grouping portions of each query block (i.e. "... the plan generation phase optimizes each subquery in the QOG independently, starting with the leaves..., enumerate join strategies and prune the search space using a branch and bound heuristic." The preceding text excerpt clearly indicates that subplans are created for each subquery/query block, and that the subplans are based on enumerating join strategies (e.g. ways of grouping portions of each query block.) (Page 3, Column 1, Paragraph 3; Column 2, Paragraph 1); determining at least one favorable access plan for each subplan of each query block, said at least one favorable access plan determined based at least in part on estimated execution costs (i.e. "Cost estimation is an integral part of the enumeration algorithm, because it is through comparing the costs of partial access plans that the ASA optimizer can quickly prune significant portions of the join strategy search space... determine the cheapest strategy and construct the detailed access plan for that strategy." The preceding text excerpt clearly indicates that at least the cheapest/on favorable access plan is recalled/determined for each subplan of each query block, where the access plan is determined to be favorable based on execution cost estimation.) (Page 3, Column 1, Paragraph 2; Column 2, Paragraph 1); generating an optimal access plan for each query block based upon said at least one favorable access plan determined for each subplan (i.e. "... determine the cheapest strategy and construct the detailed access plan for that strategy." The preceding text excerpt clearly indicates that the cheapest/optimal access plan is found for each subquery/query block which is based on the favorable access plan determined for each subplan.) (Page 3, Column 2, Paragraph 1); and constructing an optimal query execution plan based upon said optimal access plan generated for each query block (i.e. "... a subsequent set of strategies is retained only if each ones cost is not provably greater than the cost of the best join strategy discovered thus far. " The preceding text excerpt clearly indicates that the final, optimal query plan is based upon the set of optimal strategies (e.g. access plans) which were generated for each subquery/query block.) (Page 4, Column 2, Paragraph 5; Page 5, Column 1, Paragraph 1).

Under Section 102, a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in the single prior art

reference. (See, e.g., MPEP Section 2131.) As will be shown below, Bowman fails to teach each and every element set forth in Applicant's claims 1-25, 36-40, and 42-49 (as well as other claims), and therefore fails to establish anticipation of the claimed invention under Section 102.

At the outset, it should be understood that Applicant does not claim to have invented the notion of determining an optimal access plan for each "query block" of a database query. "Query blocks" themselves are naturally defined in a query (i.e., the database query received from the user) by blocks of the query that cannot be flattened. At a high level, optimizers of most modern database management systems, including that of Bowman, typically create an access plan for each query block of a database query. However, Applicant's invention includes specific elements that distinguish it from the system of Bowman as will be shown below. In particular, Applicant's invention provides for creating "subplans" inside of each query block based on grouping quantifiers used in outer joins of the query block. These subplans are then optimized independently, enabling improved query optimization.

Applicant's invention defines quantifier objects and subplan objects for each query block of a database query (Applicant's Specification, page 22, lines 2-5). Significantly, Applicant's invention provides an improved optimization methodology by defining subplans inside each query block based on grouping the tables of the outer joins. If the outer joins are nested, the subplans are also nested (Applicant's Specification, page 25, lines 9-13). Applicant's invention models the left, right and full outer joins as subplans (subplan objects) which correspond to null-supplying sides of an outer join (Applicant's Specification, page 22, lines 5-6). In this Amendment, Applicant has amended claims 1, 26, and 38 to more clearly indicate that subplans are created based on grouping quantifiers of the outer joins. These amendments to the claims are being made in an effort to clarify Applicant's invention and not for purposes of limitation of its scope.

A number of subplans (subplan objects) may be created by Applicant's invention inside a given query block which contains outer joins. Applicant's careful reading of Bowman finds no corresponding teaching of creating subplans based on grouping the tables of the outer joins. The Examiner references Bowman, at page 3, Column 1,

Paragraph 3 for the teaching of creating subplans; however, the referenced portion of Bowman states as follows: "the plan generation phase optimizes each subquery in the QOG independently, starting with the leaves" (Bowman page 3, Column 1, Paragraph 3, emphasis added). Bowman's teaching of optimization of subqueries is not analogous to the creation of subplans as described in Applicant's Specification and claims. Applicant's careful review of the balance of Bowman finds no teaching of grouping together tables involved in the null-supplying sides of the outer joins nor any teaching of how such grouping can be performed. In fact, Bowman approach's provides for use of left-deep processing trees exclusively as evidenced by the following:

**Definition 1 (Join Strategy)**

As described earlier, ASA uses left-deep processing trees exclusively; hence any join strategy  $S[1, m]$  is a linear ordering of the  $m$  quantifiers  $V_t$  in the join graph

(Bowman, page 4, column 1, paragraph 2.2, Definition 1, emphasis added).

Bowman generates a strict left-deep tree access plan for each query block and treats outer joins in the same manner as any other joins. This is not the same as Applicant's approach which provides for creating subplans inside each query block based on grouping the outer joins.

In Applicant's invention the subplans which are created inside each query block are optimized independently, enabling the search space to be inspected in smaller increments. As described below in more detail, Applicant's methodology for optimization of each subplan includes selecting access methods and join methods (i.e., physical operators implementing logical joins) as well as join order selection. A favorable access plan is determined for each subplan based on properties such as CPU cost, input/output cost, order properties, and pipeline characteristics (Applicant's specification, page 22, lines 12-26). The resulting access plans which are obtained by Applicant's invention are naturally bushy in the presence of the outer joins. In contrast, Bowman and other prior art systems typically deal with outer joins by unnesting them. For example, Bowman provides for unnesting one or more outer joins to create a larger space of valid join strategies (Bowman, page 4, column 2, paragraph 3 "Join



enumeration").

These features of creating subplans based on grouping portions of each query block and optimizing them independently are specifically referenced in Applicant's claims. For example, Applicant's claim 1 (as amended) includes the following claim limitations:

creating subplans for each query block based on grouping quantifiers used in outer joins of each query block;  
determining at least one favorable access plan for each subplan of each query block, said at least one favorable access plan determined based at least in part on estimated execution costs;

(Applicant's amended Claim 1, emphasis added)

Applicant's claims also include several other features which serve to further distinguish Applicant's invention from that of Bowman. For example, Applicant's claim 10 provides that Applicant's methodology for optimization of each subplan includes the following:

placing a candidate plan segment in the next position in a current access plan being generated, said candidate plan segment representing a particular plan node, access method and join method valid at said next position;  
evaluating the current access plan including said candidate plan segment; and  
if the current access plan is less favorable than a favorable access plan previously identified, replacing said candidate plan segment with another available candidate plan segment and repeating said evaluating substep.

(Applicant's claim 10, emphasis added)

In Applicant's invention, portions of the search space are generated as a tree (see e.g., Applicant's Fig. 6). In this tree structure, the nodes (or "plan segments") represent plan nodes together with a join method/access method pair (Applicant's Specification, page 22, lines 20-31). Applicant's invention involves enumerating not only what relations are to be added in what order, but also how they are added in that specific access methods (e.g., index) and join methods (e.g., Join Hash or Join Nested Loop) are selected as access plans are built (Applicant's Specification page 32, line 29 to page 33, line 6). Hence, Applicant's enumeration methodology differs from that of Bowman.

Bowman invention focuses on the enumeration of alternative "join strategies" which are defined in Bowman's Definition 1 (above) as a linear ordering of the quantifiers (Bowman, page 4, column 1, paragraph 2.2). In the system of Bowman, the search space is the set of all permutations of the  $n$  quantifiers of a query block (i.e., the search space size is  $n!$ ). The  $n$  quantifiers in a query block is the set of all quantifiers used in that query block, which may come from both inner joins and outer joins as the outer join are "unnested" as previously described. The join algorithm of Bowman is a simple permutation enumeration algorithm of the  $n$  quantifiers (Bowman, page 4, column 2, paragraph 3 "Join Enumeration"). Moreover, only one type of join method is used between two adjacent quantifiers (i.e., Bowman uses only one physical operator implementation for the logical operator "join", namely "Join Nested Loops"). Hence, after a permutation is generated, there is only one choice of what is the join method to be used between two tables. In the system of Bowman the index for each quantifier is chosen statically after a complete permutation of the  $n$  quantifiers is generated (Bowman, page 6, column 2, line 71 of the "Procedure: ENUMERATE"; and page 5, column 2, paragraphs 3-4, "Enumerate").

In Applicant's invention, the overall search space that is inspected is much larger than the space inspected by Bowman because the quantifiers, join methods and access methods (e.g., indexes) are enumerated. It should be noted that the term "join method" as used to describe Applicant's present invention is not the same as the terms "join strategy" and "join order" which are used in Bowman. "Join method" as used in Applicant's specification and claims refers to a physical operator implementation of the logical operator "join". Examples of physical operators implementing the logical "join" are: "Join Nested Loops", "Join Hash" (also referred to as "Hash Join"), "Sort-Merge Join", and "Join Nested Block" operators (Applicant's specification, page 29, lines 23-29; Fig. 7). Applicant's join enumeration methodology involves selecting, based on cost and other factors, a particular physical operator to use for each logical join from a number of such physical operators which are implemented in the system. In addition to selecting from among the many physical operators implementing the logical operator "join", Applicant's methodology also involves consideration of more than one access method (e.g., indexes)

as an access plan is constructed (Applicant's Specification, page 33, lines 1-15). For a query block with  $n$  quantifiers, for example, the overall size of the search space inspected in the system of Applicant's invention is  $n! \times M^{(n-1)} \times A^n$  where  $M$  is the number of physical join operators implemented in the system, and  $A$  is the number of access methods (e.g., indexes) considered for each quantifier. Recall that in Bowman's system, in contrast, the size of the search space is  $n!$  for the same  $n$  quantifiers.

Although the overall size of the search space which is inspected is larger, Applicant's methodology of grouping together the quantifiers of the outer joins in subplans benefits the enumeration process in two ways. First, it reduces the size of the search space which is inspected. For example, if a query block has 10 quantifiers and 3 of them are in the null-supplying side of outer joins, Applicant's enumeration methodology is applied to two search spaces of sizes  $8! \times M^7 \times A^8$  and  $3! \times M^2 \times A^3$ . Secondly, it makes the enumeration process faster by grouping together the null-supplying tables. For example, dependent checks may be performed only once for a subplan rather than for each individual quantifier belonging to that subplan (Applicant's Specification page 43, line 20 "FIND NEXT PLANNODES CANDIDATES" line 27). In contrast, in prior art optimizers all the outer joins are typically "unnested" as noted above, which results in some inefficiencies. For example, Bowman's "FIND CANDIDATES" method checks again and again each individual quantifier that appears in any outer joins (Bowman, page 6, col. 1, "FIND-CANDIDATES" lines 34-43).

Applicant's invention also increases efficiency by enabling earlier pruning of unfavorable plans. With Applicant's approach, pruning of an unfavorable plan can be done when only a prefix has been built (Applicant's specification, page 22, line 24 to page 23, line 9). This is possible because, unlike with Bowman and other prior art systems, Applicant's invention provides for consideration of the access methods and join methods and not just the order of joining relations. Thus, with Applicant's invention one does not have to wait to have a complete access plan to know the cost of the prefix (or partial plan) that is currently under consideration. Pruning in Bowman, in contrast, is done after the complete join strategy for a query block is built (Bowman, page 6, col. 2, "Procedure: ENUMERATE", line 74).

Applicant's invention provides for improved optimization of a database query by creating "subplans" inside each query block based on grouping quantifiers used in outer joins and optimizing these subplans independently. Applicant's optimization methodology includes examination of alternative join methods (i.e., physical join operators) and access methods (e.g., indexes) and not merely the join strategy or join order for joining relations.

All told, Bowman provides no teaching of generating subplans inside each query block of a database query in the manner described in Applicant's claims. Moreover, Bowman uses a modified permutation enumeration methodology for determining a "join strategy" or order of joining relations while Applicant's invention considers particular access methods and join methods as access plans are constructed. Therefore, as Bowman does not teach or suggest all of the claim limitations of Applicant's claims 1-25, 36-40, and 42-49 (and other claims) it is respectfully submitted that the claims distinguish over this reference and overcome any rejection under Section 102.

**B. Section 103(a) rejection: Bowman in view of Du**

The Examiner has rejected claims 26-35 and 41 under 35 U.S.C. 103(a) as being obvious over Bowman in view of US Patent No. 5,649,591 (which the Examiner clarified by phone was meant to refer to US Patent No. 5,694,591) of Du et al (hereinafter "Du"). The Examiner acknowledges that Bowman fails to disclose generating a bushy execution tree based upon the optimal access plan for each query block and adds Du for these teachings. The Examiner states that it would have been obvious to one skilled in the art at the time of Applicant's invention to modify the teachings of Bowman to include the teachings of Du with the motivation to optimize data retrieval from a multi-database system by restructuring a query to optimize database response time.

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under this section, the Examiner must establish: (1) that there is

some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). As will be shown below, the references cited by the Examiner fail to meet the requisite condition of teaching or suggesting all of Applicant's claim limitations.

The Examiner relies on Bowman as substantially teaching the claimed invention (as per the Examiner's rejection under Section 102 above), but acknowledges that Bowman fails to disclose generating a bushy execution tree based upon the optimal access plan determined for each query block. The Examiner adds Du for this teaching of generating a bushy execution tree.

A processing tree is called a "bushy tree" if it has composite (i.e., not necessarily base relations) for the left and right children of the join nodes (Applicant's Specification, page 9, lines 4-5). The access plans which are obtained by Applicant's invention are naturally bushy in the presence of the outer joins as a result of Applicant's methodology for grouping together tables used in outer joins into "subplans" and optimizing these subplans independently (Applicant's Specification, page 31, lines 26-31). As discussed above, Bowman does not teach creating "subplans" inside query blocks and optimizing these subplans independently as described in Applicant's Specification and claims. Du does not cure these deficiencies of Bowman. Applicant's review of Du finds that it contains no teaching of grouping together tables used in outer joins of a query.

In addition, Du's approach differs from Applicant's invention in that Du provides for a two-phase (or two-step) optimization process. The first phase of Du's query optimization process involves use of left-deep trees exclusively, which is distinguishable from Applicant's invention for the reasons previously described in response to the Section 102 rejection. In its second phase, Du's system attempts to transform the left-deep join tree structure generated in phase 1 into a more balanced bushy tree structure. Du's transformation methodology includes evaluating various transformations and identifying ones which are both valid and cost improving (Du, col. 10, lines 17-21).

Applicant's invention provides a single unified methodology in which costs of partial access plans are compared as plans are built. Thus, Applicant's approach enables early identification and pruning of unfavorable plans (Applicant's Specification, page 21, lines 6-12). Du's two-phase approach, however, requires additional processing after a complete access plan is generated in an effort to transform it into a more balanced structure.

Du's invention is also implemented in a multidatabase system (MDBS) and is focused on reducing query response time in retrieving data from a plurality of databases on a distributed network (Du Abstract, Claim 1, Claim 6). In the MDBS system of Du, the bushy or "balanced" tree is created to address the different cost of retrieving relations from distributed databases, where each relation may potentially be in a different distributed database. The balanced tree which is created by Du's two-phase optimization does not correspond with the original query. In contrast to Applicant's approach of creating subplans based on grouping the outer joins, in Du's system the outer joins are treated as any other join as the first phase of Du's optimization process generates a left deep join query structure (Du, col. 7, lines 47-53; Fig. 1). The bushy (or balanced) tree is hence created artificially in the second phase of Du's optimization process based on the distribution of the relations in different distributed databases; the bushy tree has no relationship to the structure of the original query.

Accordingly, as the prior art reference(s), even when combined, fail to teach or suggest all the claim limitations, it is respectfully submitted that Applicant's claimed invention as set forth by these claims is distinguishable over the two references, and that the rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

#### Conclusion

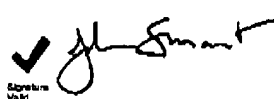
In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that

the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: May 26, 2006

 Digitally signed  
by John A.  
Smart  
Date:  
2006.05.26  
16:05:58 -0700  
Signature  
Valid

John A. Smart, Reg. No. 34,929  
Attorney of Record

408 884 1507  
815 572 8299 FAX